MCP41HV51 Digital Potentiometer (Arduino)

8-bit MCU / Electronics 2017. 12. 28. 20:04

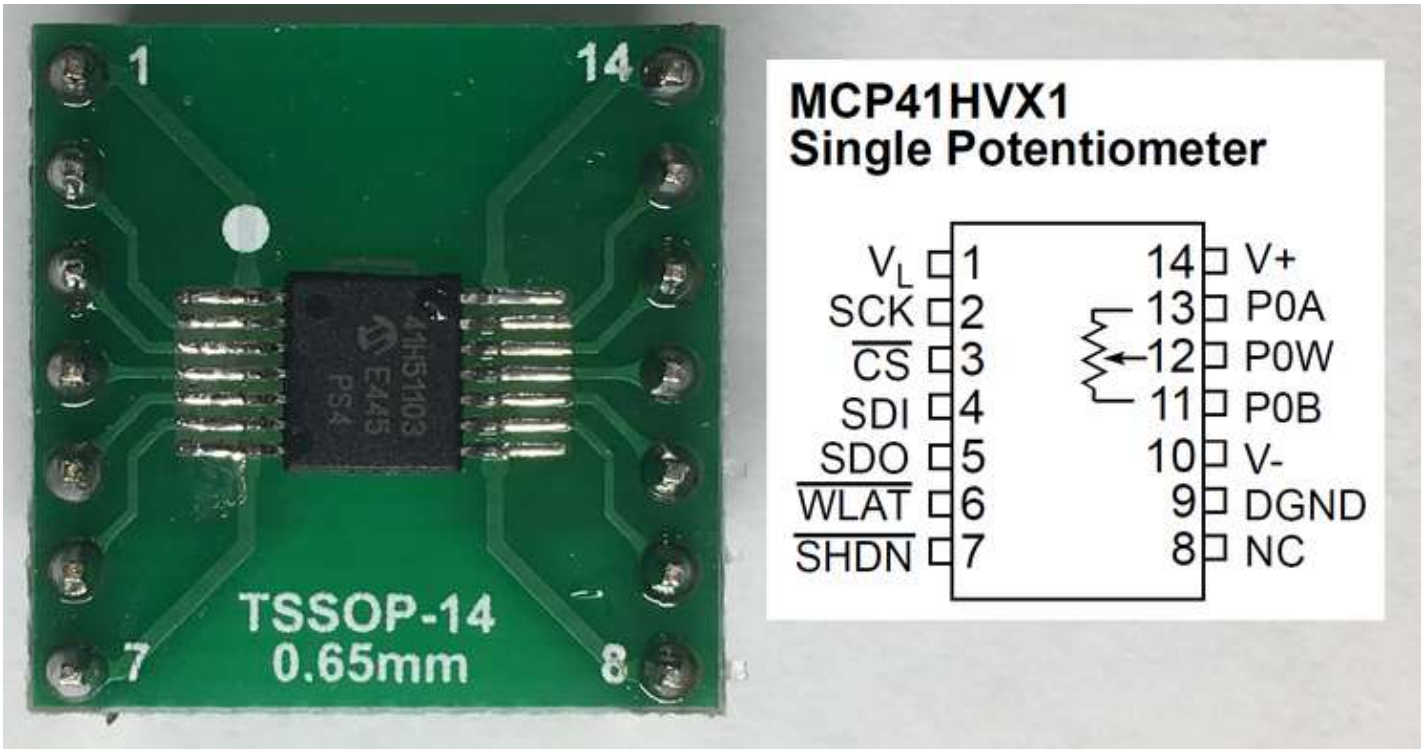
The resistance composed of the electronic circuit plays an important role in the operation of the circuit. When the circuit is designed to change the operation by adjusting the resistance value, a variable resistor called a potentiometer can be used. In general, variable resistance can be changed by turning the dig in the shape of a handle or a screwdriver. There is also a variable resistor that can change the resistance value by applying a digital signal instead of a physical method. This is called a digital potentiometer, and it is often used because it has the advantage of controlling the resistance value with a microcontroller (MCU).

However, the disadvantage of digital variable resistors is that the operating voltage and current are greatly limited. So, in this article, we will look at Microchip's MCP41HV51 model that can be operated at high voltage among digital variable resistors. This product can be used in the range of up to + 36V or ± 18V voltage, so it is good for use as an amplification factor control in the OP Amp circuit. The Arduino Mega was used to test the MCP41HV51 digital variable resistor.

1. Features of the MCP41HV51 digital variable resistor

The picture below shows the MCP41HV51 digital variable resistor in the TSSOP-14 package. The MCP41HV51 model is only sold in TSSOP and QFN packages, not in DIP packages, so it must be soldered to the converter board for use on breadboards or universal boards. The exact model name of the product I purchased is MCP41HV51-103, with 103 on the back, indicating that the resistance is 10 kΩ. Other products have resistance values of 5kΩ, 50kΩ and 100kΩ.

MCP41HVX1 is indicated on the pin map on the right, and the MCP41HV51 model with 5 attached to X has 8 bit resolution. The MCP41HV31 model with the same operation method has 7 bits of resolution and is cheaper. Looking at the SCK, / CS, SDI, and SDO pins, you can see that this digital variable resistor communicates in SPI. Pins P0A and P0B are connected at both ends of the internal resistor, and a wiper is also connected to pin P0W.



The role of each pin of the MCP41HV51 digital variable resistor is as follows. The power supply voltage of the digital circuit is input between the VL pin and DGND pin, and the analog power supply voltage that operates the internal resistance is input between the V + and V- pins. If the low state is input to the / WLAT pin, the value received through SPI communication is transferred to the wiper register to change the resistance value. If the input state of the / SHDN pin is low, the position of the wiper is maintained, but the connection between the P0A, P0W, and P0B pins and the internal resistor is disconnected and literally shuts down.

TABLE 3-1: PINOUT DESCRIPTION FOR THE MCP41HVX1

Pin				Function
TSSOP 14L	Symbol	Type	Buffer Type	
1	V_L	P	—	Positive Digital Power Supply Input
2	SCK	I	ST	SPI Serial Clock pin
3	$\overline{CS}$	I	ST	Chip Select
4	SDI	I	ST	SPI Serial Data In pin
5	SDO	O	—	SPI Serial Data Out
6	$\overline{WLAT}$	I	ST	Wiper Latch Enable 0 = Received SPI Shift Register Buffer (SPIBUF) value is transferred to Wiper register 1 = Received SPI data value is held in SPI Shift Register Buffer (SPIBUF)
7	$\overline{SHDN}$	I	ST	Shutdown
8	DGND	P	—	Ground
9	NC	—	—	Pin not internally connected to die. To reduce noise coupling, connect pin either to DGND or V_L.
10	V-	P	—	Analog Negative Potential Supply
11	P0B	I/O	A	Potentiometer 0 Terminal B
12	P0W	I/O	A	Potentiometer 0 Wiper Terminal
13	P0A	I/O	A	Potentiometer 0 Terminal A
14	V+	P	—	Analog Positive Potential Supply

Legend: A = Analog, ST = Schmitt Trigger, I = Input, O = Output, I/O = Input/Output, P = Power

The digital power supply voltage indicated by VL and DGND can be located anywhere in the analog power supply voltage range indicated by V + and V-, so it is convenient to freely match the levels of the two power supply voltages. The voltage on the V + pin can be as high as 36V above the voltage on the V-pin, but the minimum voltage is 10V. So, if the range of analog power supply voltage is less than 10V, it seems to be difficult to use. Either digital power supply or analog power supply may be applied first, so you do not have to worry about the order of power supply.

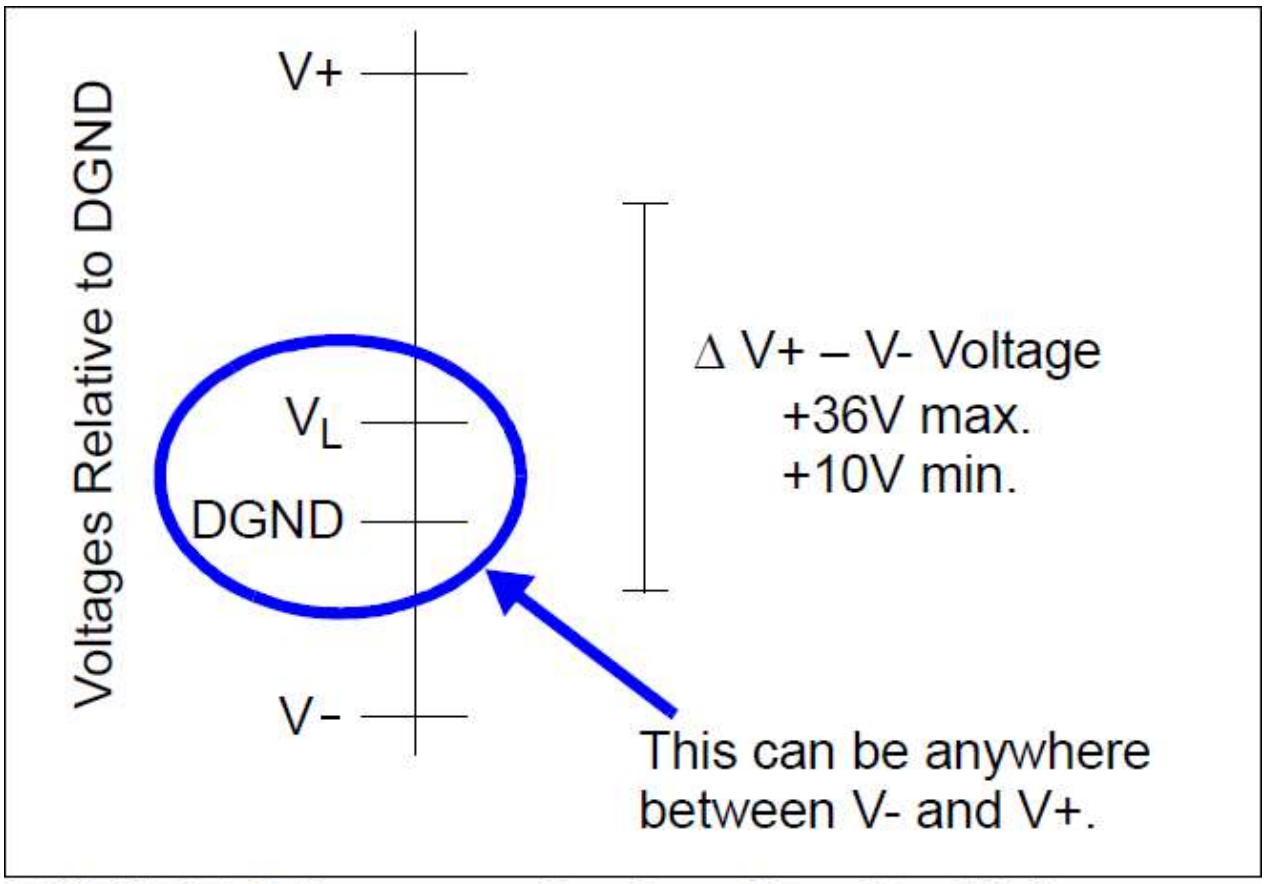


FIGURE 5-9: *Analog Circuitry Voltage Ranges.*

2. Register

Let's look at the resistor to control the MCP41HV51 digital variable resistor. First, there is a Wiper0 register that determines the location of the wiper. The wiper position is located between the R0A pin and R0B pin in proportion to the value of the Wiper0 register. When the value of the Wiper0 register is changed, the resistance between the R0W pin and R0A to which the wiper is connected and the R0W pin and R0B pin changes. So, to change the resistance value, you can change the value of the Wiper0 register. Since the resolution of this digital variable resistor is 8 bits, the range of the Wiper0 register is 0x00 to 0xFF (255). When reset, the initial value of Wiper0 is the middle value, 0x7F.

The other register has a TCON0 register, which stands for Terminal control. In the description below, the lower 4 bits of the TCON0 register are used to control the digital variable resistor. One bit determines whether to shut down software, and the other three bits determine whether to connect the R0A, R0W, and R0B pins to internal resistors. When this digital variable resistor is reset, it is initialized to 0xFF, and the software does not shut down, but R0A, R0W, R0B pins are connected to the internal resistor. So I don't need to touch the TCON0 register for general use. When a low state is input to the / SHDN pin, this digital variable resistor is shut down regardless of the value of the R0HW bit.

REGISTER 4-1: TCON0 BITS(1)

R-1	R-1	R-1	R-1	R/W-1	R/W-1	R/W-1	R/W-1
D7	D6	D5	D4	R0HW	R0A	R0W	R0B
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-4 **D7-D4:** Reserved. Forced to "1"

bit 3 **R0HW:** Resistor 0 Hardware Configuration Control bit
This bit forces Resistor 0 into the "shutdown" configuration of the Hardware pin
1 = Resistor 0 is **not** forced to the hardware pin "shutdown" configuration
0 = Resistor 0 is forced to the hardware pin "shutdown" configuration

bit 2 **R0A:** Resistor 0 Terminal A (P0A pin) Connect Control bit
This bit connects/disconnects the Resistor 0 Terminal A to the Resistor 0 Network
1 = P0A pin is connected to the Resistor 0 Network
0 = P0A pin is disconnected from the Resistor 0 Network

bit 1 **R0W:** Resistor 0 Wiper (P0W pin) Connect Control bit
This bit connects/disconnects the Resistor 0 Wiper to the Resistor 0 Network
1 = P0W pin is connected to the Resistor 0 Network
0 = P0W pin is disconnected from the Resistor 0 Network

bit 0 **R0B:** Resistor 0 Terminal B (P0B pin) Connect Control bit
This bit connects/disconnects the Resistor 0 Terminal B to the Resistor 0 Network
1 = P0B pin is connected to the Resistor 0 Network
0 = P0B pin is disconnected from the Resistor 0 Network

Note 1: These bits do not affect the wiper register values.
2: The hardware $\overline{\text{SHDN}}$ pin (when active) overrides the state of these bits. When the $\overline{\text{SHDN}}$ pin returns to the inactive state, the TCON register will control the state of the terminals. The $\overline{\text{SHDN}}$ pin does not modify the state of the TCON bits.

3. Signal of SPI communication

To change the value of the register we looked at above, we need to send a signal to the MCP41HV51 IC via SPI communication. The communication mode required by this IC is (CPOL, CPHA) = (0, 0), (1, 1), which corresponds to modes 0 and 3. The SPI frequency should be 10MHz or less when the digital power supply voltage (VL) is 2.7V or higher, and data must be transmitted according to the format below.

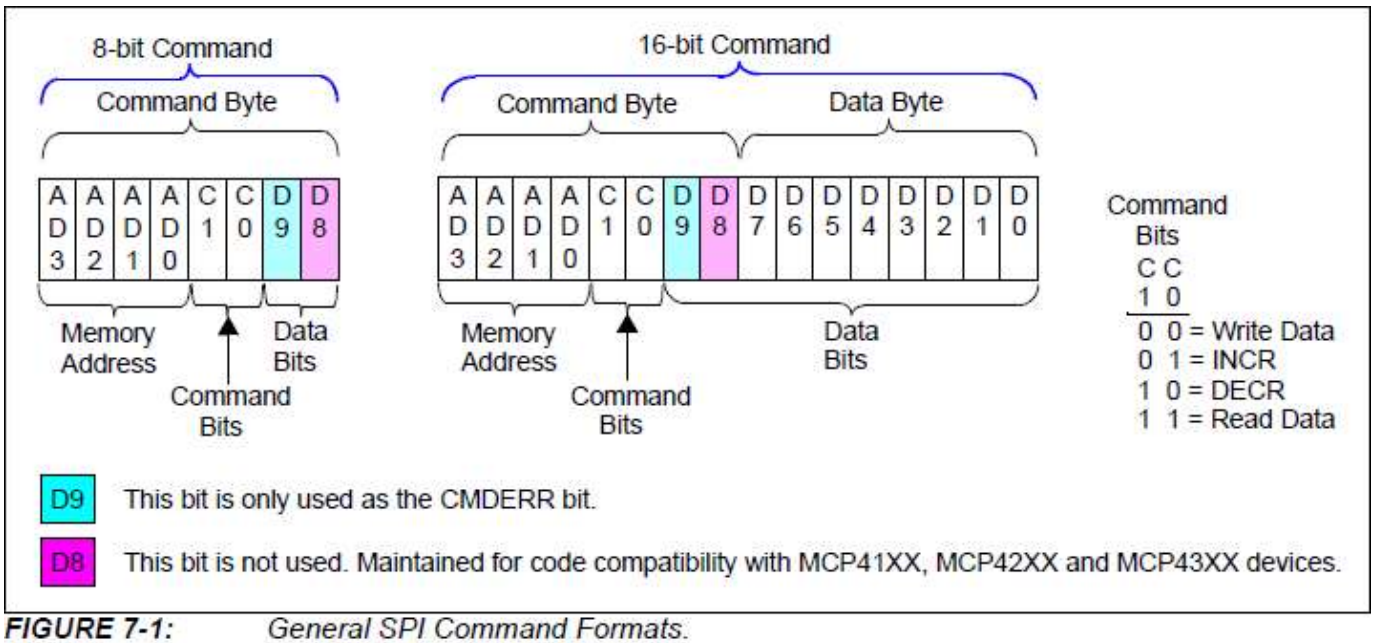


FIGURE 7-1: *General SPI Command Formats.*

The table below details the data to be sent on the MOSI signal line and the data to be received on the MISO signal line. You can send a signal to change the value of the Wiper0 or TCON0 register with the MISO signal line. If you do not need to check the current Wiper0, TCON0 register value or CMDERR bit, you do not need to use the MISO signal line.

TABLE 7-2: MEMORY MAP AND THE SUPPORTED COMMANDS

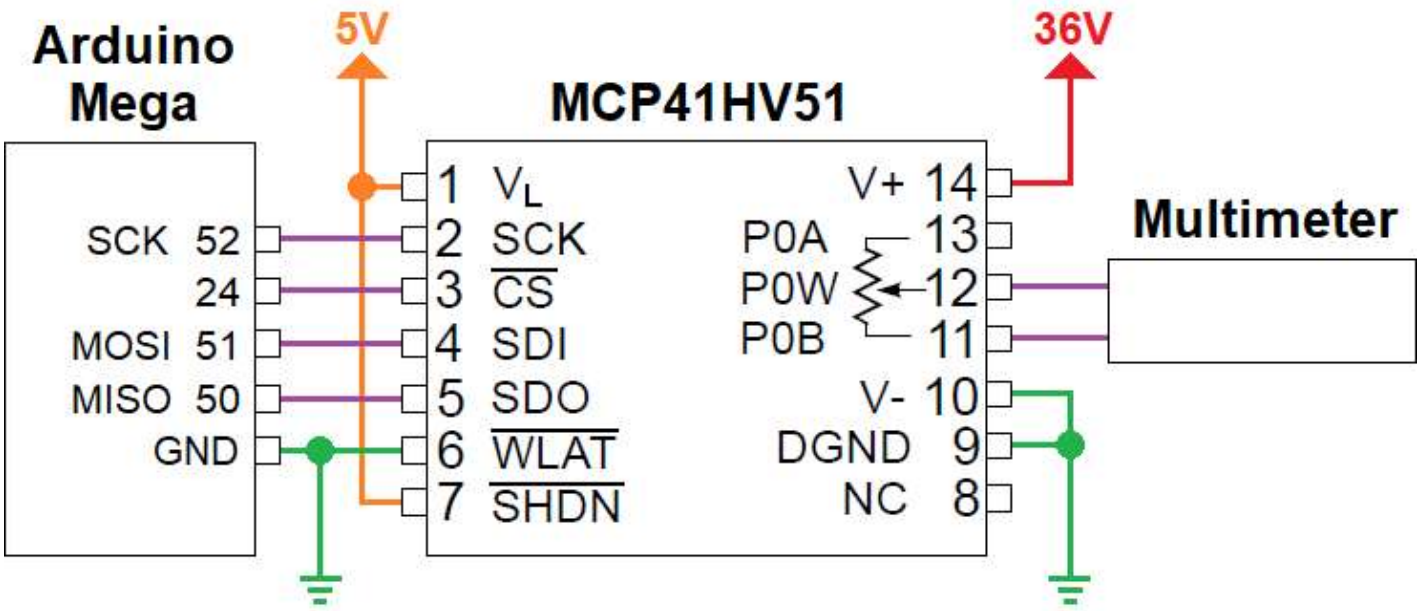
Address		Command	Data (10-bits)(1)	SPI String (Binary)	
Value	Function			MOSI (SDI pin)	MISO (SDO pin)(2)
00h	Volatile Wiper 0	Write Data	nn nnnn nnnn	0000 00nn nnnn nnnn	1111 1111 1111 1111
		Read Data	nn nnnn nnnn	0000 11nn nnnn nnnn	1111 111n nnnn nnnn
		Increment Wiper	—	0000 0100	1111 1111
		Decrement Wiper	—	0000 1000	1111 1111
01h – 03h(4)	Reserved	—	—	—	—
04h(3)	Volatile TCON Register	Write Data	nn nnnn nnnn	0100 00nn nnnn nnnn	1111 1111 1111 1111
		Read Data	nn nnnn nnnn	0100 11nn nnnn nnnn	1111 111n nnnn nnnn
05h – 0Fh(4)	Reserved	—	—	—	—

Note 1: The data memory is eight bits wide, so the two MSbs (D9:D8) are ignored by the device.
2: All these address/command combinations are valid, so the CMDERR bit is set. Any other address/command combination is a command error state and the CMDERR bit will be clear.
3: Increment or Decrement commands are invalid for these addresses.
4: **Reserved addresses**: Any command is invalid for these addresses.

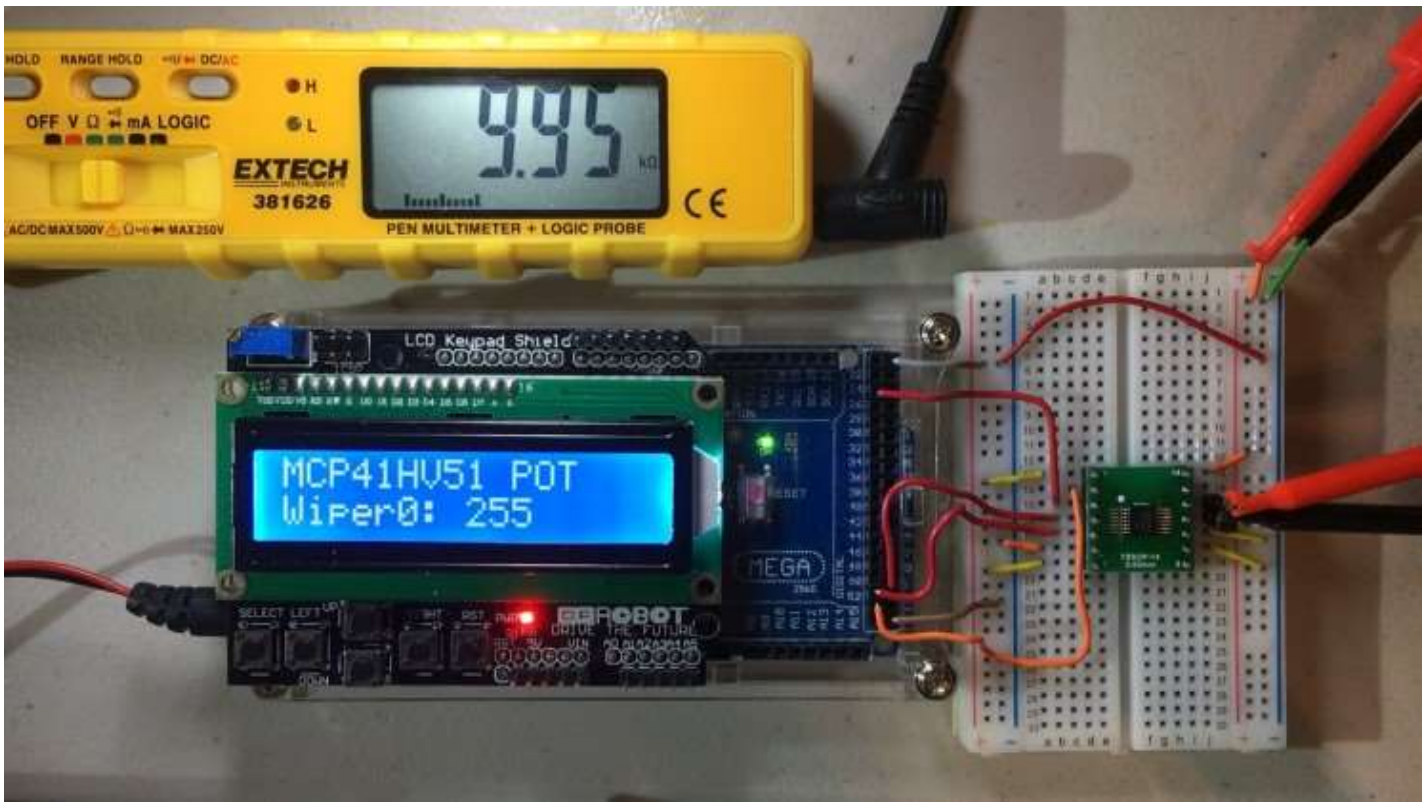
4. Circuit Composition

Now that you have confirmed the MCP41HV51's pinmap, register to operate, and SPI communication method, let's configure the circuit to control with Arduino. Since the Arduino communicates at a 5V logic level, 5V is input to the digital power pin VL pin. Also, I connected the SPI pin of the

MCP41HV51 to the SPI communication pin of the Arduino Mega, and the / CS pin was connected to pin 24 to control the CS (SS) signal in software. The / WLAT pin is connected to GND so that the value of the received wiper register is reflected immediately, and the / SHDN pin is connected to 5V to prevent shutdown. Between the V + pin and the V- pin, 36V, the maximum voltage, was applied by connecting two power supplies in series. The resistance was measured by connecting a multimeter probe between the P0W and P0B pins.



The photo below shows how the circuit is structured as above, and when the code is uploaded to the Arduino Mega. The keypad LCD shield is used to easily change the Wiper0 register to a button and display the set value. When the Wiper0 resistor is the maximum value of 255, it points to 9.95kΩ, which is quite close to the 10kΩ resistance on the specifications.



5. Operation video

You can see in the video that the resistance value between the P0W pin and PB pin changes when you operate the Wiper0 resistor. Initially, when the value of the Wiper0 resistor is 0, the 119Ω resistance value measured is the minimum resistance of the wiper, which fits within the range indicated in the datasheet. When the value of the Wiper0 register increases, the resistance value increases almost proportionally. By operating the Wiper0 register, the resistance value of the digital variable resistor can be controlled.

6. Code uploaded on Arduino Mega

The code uploaded to the Arduino Mega to control the MCP41HV51 digital variable resistor is shown below. I wrote the code to change the value of the Wiper0 variable when the key is pressed, display it on the LCD, and send a command via SPI communication.

```
#include <LiquidCrystal.h>
#include <SPI.h>

LiquidCrystal lcd ( 8 , 9 , 4 , 5 , 6 , 7 ); // select the pins used on the LCD panel

int lcd_key1 = 0 ;     // 1st button reading
int lcd_key2 = 0 ;     // 2nd button reading
int adc_key_in = 0 ;

int TCON0;            // TCON register
int Wiper0 = 127 ;     // Wiper register
int Wiper0_pre = 127 ; // Previous Wiper register

#define btnRIGHT 0
#define btnUP 1
#define btnDOWN 2
#define btnLEFT 3
#define btnSELECT 4
#define btnNONE 5


// read button
int read_LCD_buttons ()
{
    adc_key_in = analogRead ( 0 ); // read ADC value

    if (adc_key_in < 50 )
        return btnRIGHT;
    if (adc_key_in < 175 )
        return btnUP;
    if (adc_key_in < 330 )
        return btnDOWN;
    if (adc_key_in < 520 )
        return btnLEFT;
    if (adc_key_in < 830 )
        return btnSELECT;

    return btnNONE;
}


void setup ()
{
```

```
    lcd. begin ( 16 , 2 );    // Initialize LCD
    lcd. setCursor ( 0 , 0 ); // (0, 0) position
    lcd. print ( "MCP41HV51 POT" );

    lcd. setCursor ( 0 , 1 ); // (0, 0) position
    lcd. print ( "Wiper0:" );

    pinMode ( 24 , OUTPUT );    // Pin 24 OUTPUT, Slave Select (SS)
    digitalWrite ( 24 , HIGH ); // SS High
    SPI. begin ();              // Initialize SPI (Default: 4MHz, Mode 0)
}


void loop ()
{
    int digit;
    int i;

    while ( 1 )
    {
        lcd_key1 = read_LCD_buttons ();
        delay ( 10 );   // 10ms delay
        lcd_key2 = read_LCD_buttons ();
        if (lcd_key1 == lcd_key2)  // if two button readings are same
        {
            break ;
        }
    }

    lcd. setCursor ( 0 , 1 );  // (0, 1) position

    switch (lcd_key1)
    {
    case btnRIGHT:
        Wiper0 = Wiper0 + 10 ;
        break ;
    case btnLEFT:
        = Wiper0 Wiper0 - 10 ;
        break ;
    case btnUP:
        Wiper0 ++;
        break ;
    case btnDOWN:
        Wiper0--;
        break ;
    case btnSELECT:
        break ;
    case btnNONE:
        break ;
    }

    if (Wiper0> 255 )
    {
        Wiper0 = 255 ;
    }
    else if (Wiper0 < 0 )
    {
```

```
        Wiper0 = 0 ;
    }

    // Find digit of Wiper0
    if (Wiper0> 99 )
    {
        digit = 3 ;
    }
    else if (Wiper0> 9 )
    {
        digit = 2 ;
    }
    else
    {
        digit = 1 ;
    }

    // display Wiper value (Wiper0)
    lcd. setCursor ( 8 , 1 );  // (8, 1) position

    for (i = 3 ; i> digit; i--) // space
    {
        lcd. print ( "" );
    }

    lcd. print (Wiper0); // Wiper value

    if (Wiper0! = Wiper0_pre) // If Wiper value is changed
    {
        // SPI communication
        digitalWrite ( 24 , LOW );
        SPI. transfer ( 0x00 );    // Write Data to Wiper0 register (address: 0x0)
        SPI. transfer (Wiper0);  // Value of Wiper0 register
        digitalWrite ( 24 , HIGH );
        Wiper0_pre = Wiper0;
    }

    delay ( 300 );
}
```

The digital variable resistor of MCP41HV31 or MCP41HV51 model can be controlled in this way. You can control it with AVR ATmega128 or STM32 MCU instead of Arduino. The data sheet of this digital variable resistor is large enough to be close to page 70 except for the Appendix, and there are many other contents besides the description of this article. For detailed operating conditions, resistance errors, temperature characteristics, allowable current graphs, and application circuits, see the datasheet.